(12) UK Patent Application (19) GB (11) 2 216 307 (13) A

(43) Date of A publication 04.10.1989

(21) Application No 8903962.2

(22) Date of filing 22.02.1989

(30) Priority data
(31) 162738 (32) 01.03.1988 (33) US

(51) INT CL⁴
G06F 12/06

(52) UK CL (Edition J)
G4A AMB

(56) Documents cited
EP 0261751 A EP 0251559 A EP 0224691 A
EP 0165822 A EP 0024288 A

(58) Field of search
UK CL (Edition J) G4A AMB
INT CL⁴ G06F

(71) Applicant
Ardent Computer Corporation

(Incorporated in the USA - California)

880 West Maude Avenue, Sunnyvale, California 94086, United States of America

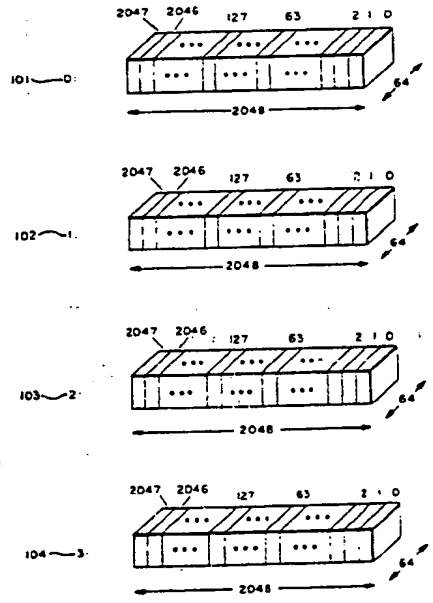
(72) Inventors
Glen S. Miranker
Steve Johnson

(74) Agent and/or Address for Service
Potts Kerr and Co
15 Hamilton Square, Birkenhead, Merseyside, L41 6BR, United Kingdom

(54) Vector register file

(57) A vector register file uses static random access memories organized in a plurality of banks to provide a logical multi-ported vector register file. Each of the four banks may be addressed independently during the same clock cycle. Further, the vector register file is divided into a plurality of context areas, each context area usable by a separate process. Use of a plurality of context areas avoids problems of swapping context data into and out of the vector register file. The method for addressing the vector register file allows for optional addressability to individual cells of the vector register file.

FIG_1



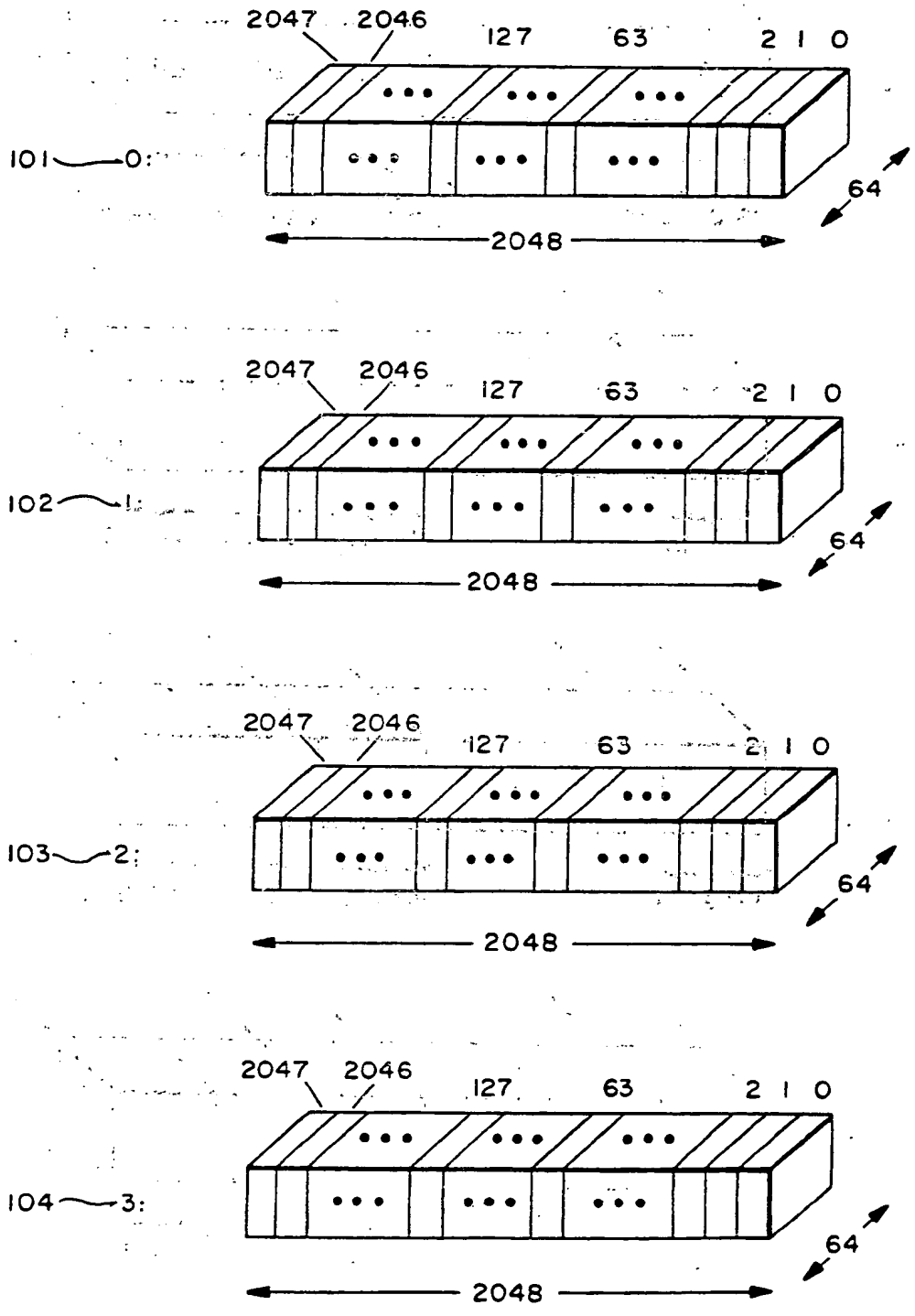
BEST AVAILABLE COPY

GB 2 216 307 A

At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

FIG_1

2216307



3NSDOCID <GB _ 2216307A_I >

FIG_2

2216307

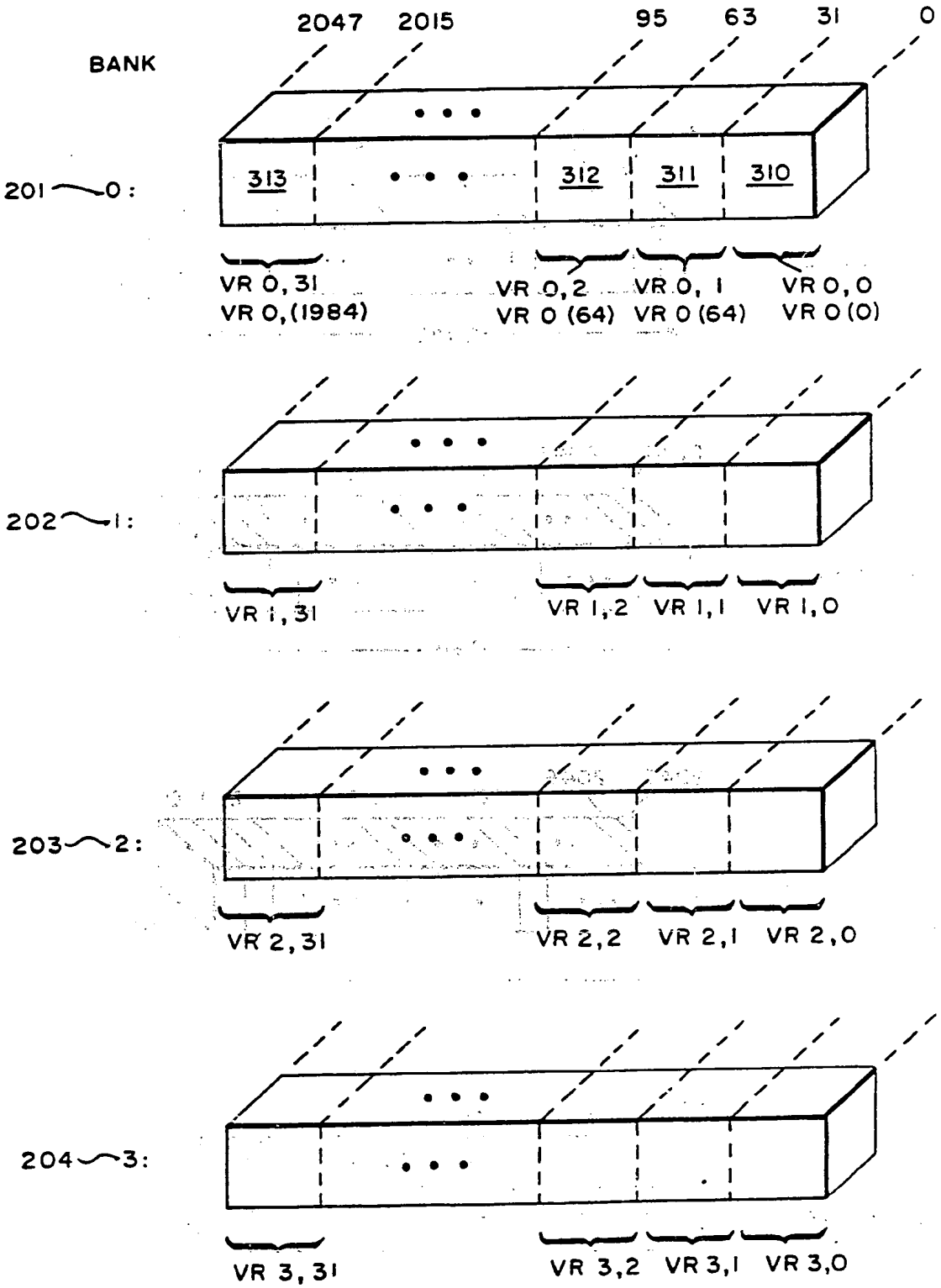
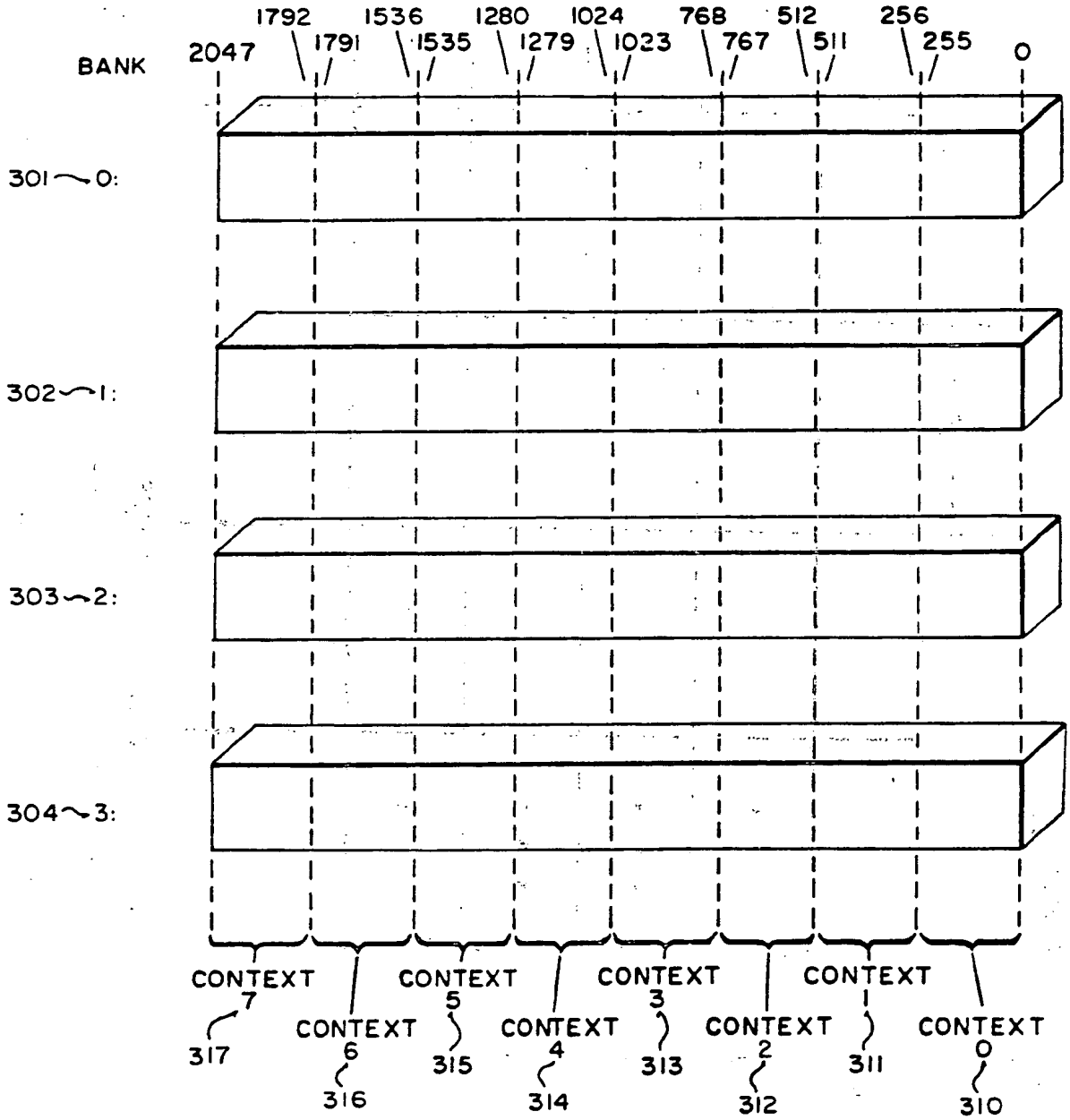
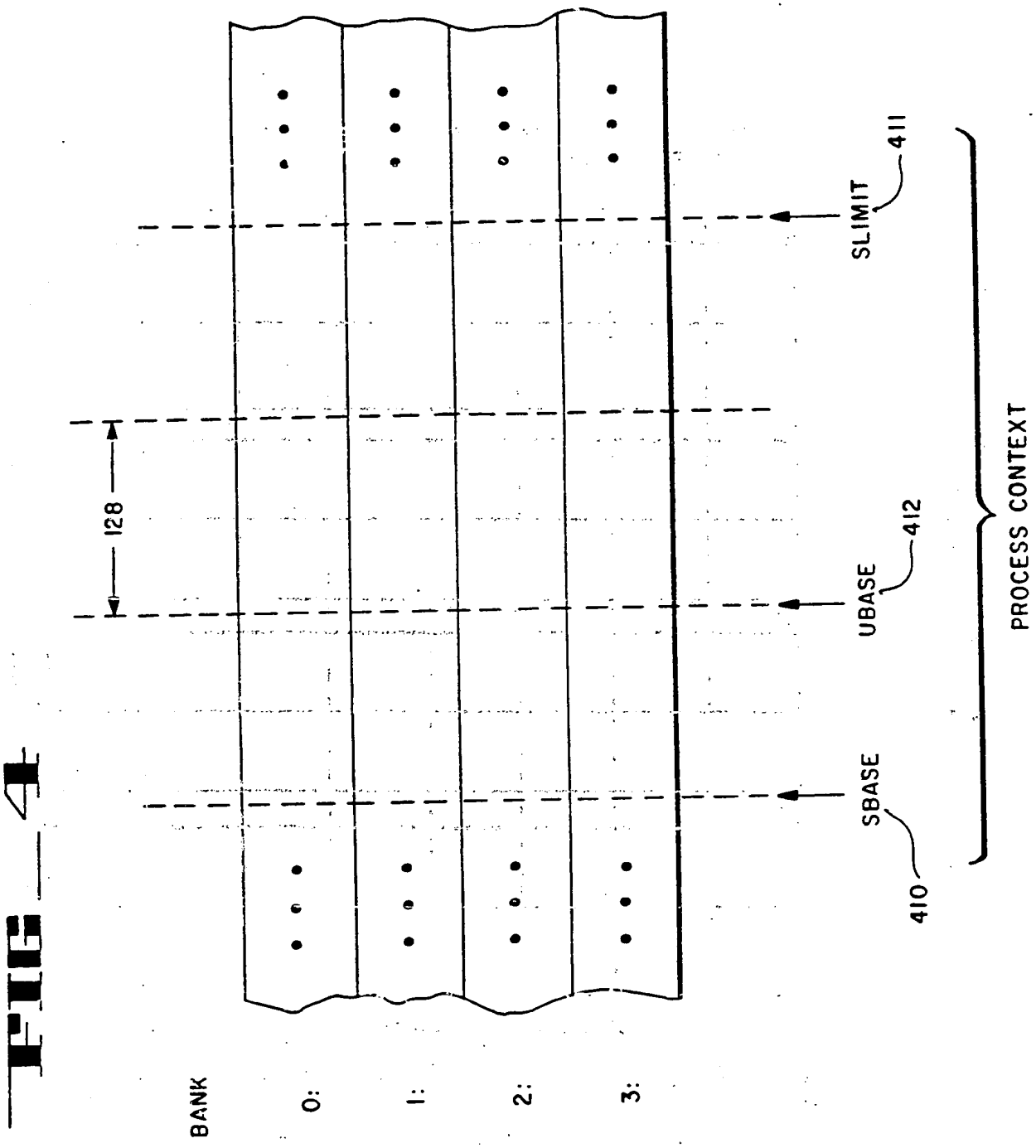


FIG 3



BNSDOCID: <GB___2216307A__I_>

FIG—4


BANK
0:
1:
2:
3:
128
SLIMIT
411
UBASE
412
SBASE
410
PROCESS CONTEXT

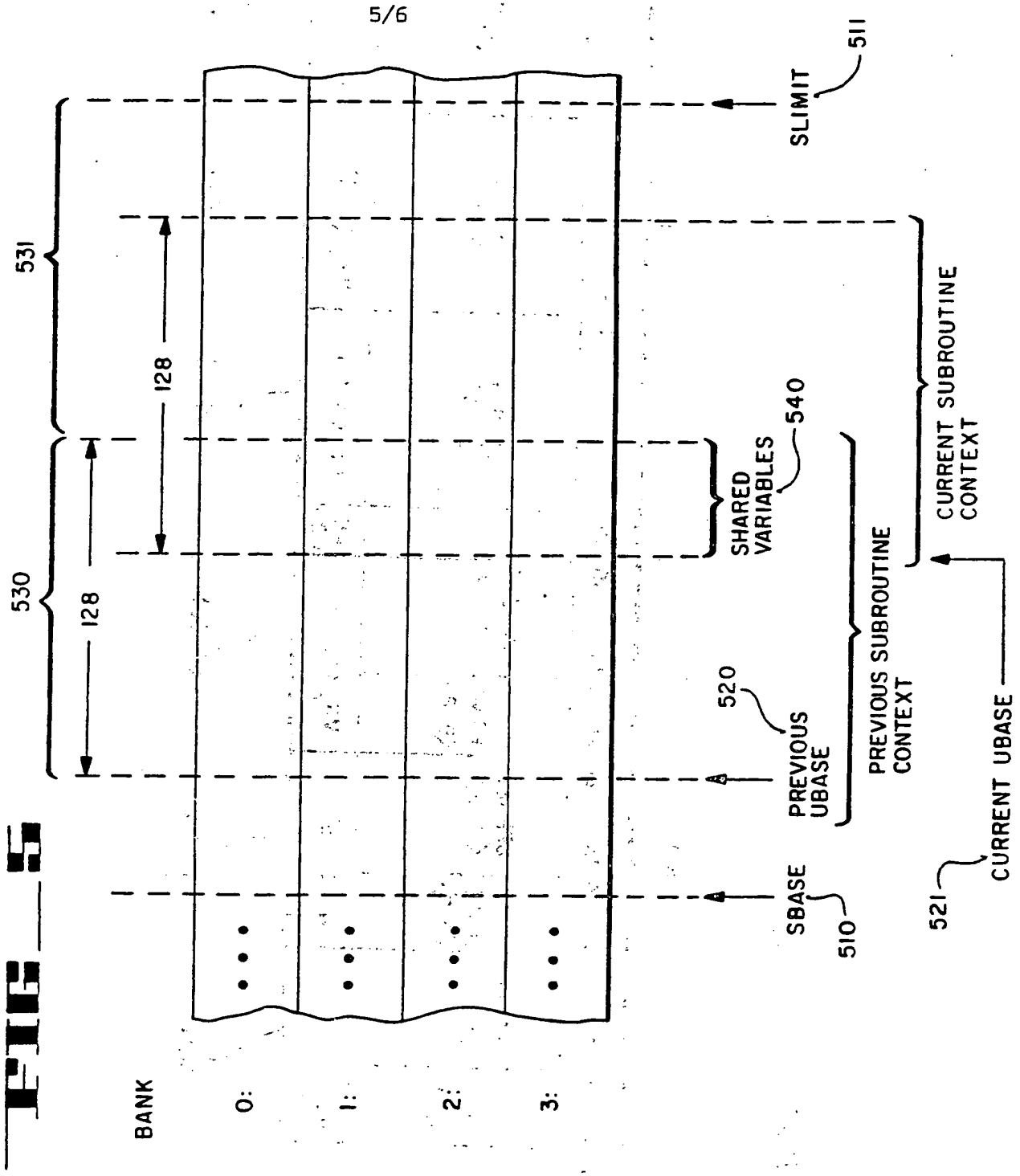
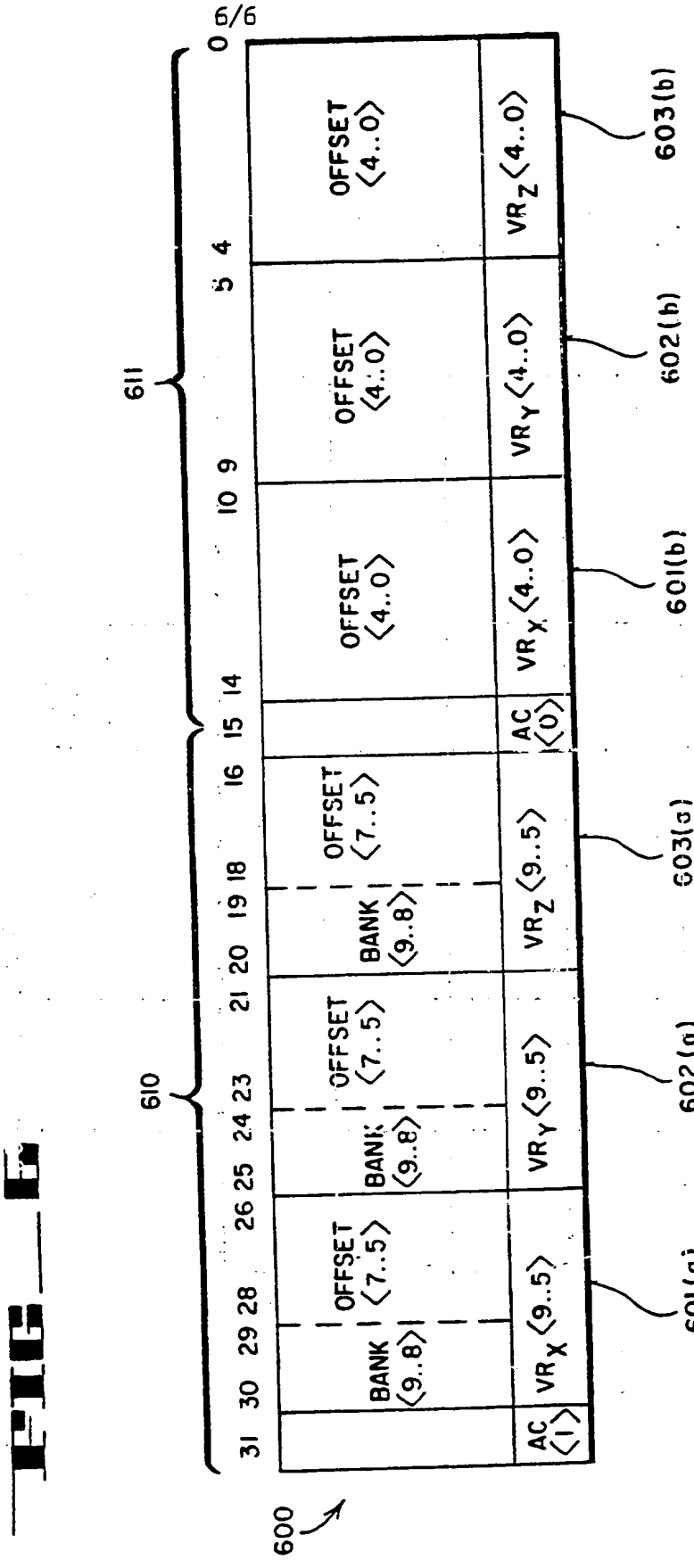


FIG_5

INSDOCID: <GB___2216307A__I_>

2216307

FIG. 6



2216307

VECTOR REGISTER FILE

BACKGROUND OF THE INVENTION

1. Field of the Invention.

The present invention relates to vector register files for vector processing computer systems.

2. Prior art.

In a vector processing computer system, a vector register file is typically used for storing vectors for computations. Operations in such a system may then be carried out on a vector at a time. For example, a operation such as ADD VREG1, VREG2, VREG3 in a vector processing system may cause the first element in vector register 1 to be added to the first element of vector register 2 and the result to be stored in the first element of vector register 3. The second element of vector register 1 would, likewise, be added to the second element of vector register 2 and the result would be stored in the second element of vector register 3. Each element of vector register 1 and vector register 2 is similarly processed and stored in the corresponding element of vector register 3.

To increase processing speed in such computer systems, multi-ported memories are commonly used to construct the vector register files. Such multi-ported memories offer the advantage of being able to read or write to multiple memory locations during a single memory cycle. However, such multi-ported memories suffer from several drawbacks including typically being more expensive to design, construct and purchase than single-ported memories, being less readily available

3NSDOCID <GB___2216307A__I_>

than single-ported memories and typically being slower than single-ported memories.

Therefore, as one objective of the present invention, it is desired to design a vector register file for a vector processing computer system which utilizes single-ported memory cells, while offering the speed advantages of multi-ported memories.

Known vector processing systems typically employ vector register files of a somewhat limited size. For example, a typical CRAY computer system manufactured by Cray Research of Minneapolis, Minnesota may have 8 vector registers, each with a depth of 64 elements for a total of 512 elements. Other known systems typically employ between 2 and 4 times as much vector register space. Normally, the increased vector register space is accomplished by increasing the depth (i.e. to a depth of 128 or 256) of each vector register. As a second objective of the present invention, it is desired to increase the available amount of vector registers space and to provide increased flexibility in the utilization and management of that space.

A third objective of the present invention is to provide programming flexibility allowing for execution of instructions which operate on vector, and computations which require access to a vector at an arbitrary point. Examples of such computations arise in recurrence and/or cumulation calculations.

SUMMARY OF THE INVENTION

A vector register file and methods for managing access to the vector register file is described. The present invention discloses use of static random access memories (SRAMs) organized in a plurality of banks and operating at least two times the system clock speed to simulate multi-ported memories. Through use of SRAMs to build a vector register file a number of inventive features are achieved.

Use of SRAMs allows a computer system to be configured with a relatively large amount of vector register file space. This space may be divided into a plurality of context areas, each context area for supporting a separate process. The use of such context areas avoids the need to swap the context information in a vector register file to disk when switching between processes. The present invention discloses use of a system base register, a user base register and a system limit register to support the use of a plurality of context areas.

The present invention further discloses an addressing scheme for addressing the context areas which allows for optional addressability to individual elements or cells in a vector.

NSDOCID <GB___2216307A__I_>

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram illustrating banks of memories as may be utilized by the present invention.

Figure 2 is a block diagram illustrating a method of dividing the banks of memories into a plurality of vector registers as taught by the present invention.

Figure 3 is a block diagram illustrating a method of dividing the banks of memories into a plurality of context areas as taught by the present invention.

Figure 4 is a block diagram illustrating the use of system base, user base and system limit registers taught by the present invention.

Figure 5 is a block diagram illustrating the use of system base, user base and system limit registers and the use of shared variables between subroutines taught by the present invention.

Figure 6 is a block diagram illustrating an operand descripter as may be utilized by the present invention.

DETAILED DESCRIPTION OF THE PRESENT INVENTION

A vector register file is described. In the following description, numerous specific details are set forth such as specific number of bits, specific dimensions, etc., in order to provide a thorough understanding of the present invention. It will be obvious, however, to one of ordinary skill in the art that the present invention may be practiced without these specific details. In other instances, well-known circuits, structures and techniques have not been shown in detail in order to avoid unnecessarily obscuring the present invention.

VECTOR REGISTER FILE

The present invention discloses arranging a vector register file in a vector processing computer system as a plurality of banks of random-access memories (RAMs). For the purposes used herein, the term "bank" shall refer to an independently cycleable collection of memories. That is, each of the plurality of banks may be independently addressed during the same clock cycle.

The preferred embodiment of the present invention will be described in more detail with reference to Figure 1. The preferred embodiment utilizes four banks of memories, 101, 102, 103 and 104, for its vector register file. Each bank, 101, 102, 103 and 104 comprises eight 2,048 x 8 static random-access memories (SRAMs) arranged such that a bank may be comprised of 2,048 64-bit elements.

INSDOCID <GB _ 2216307A_ I_>

The present invention achieves a logical equivalent of an 8-ported memory through the combination of two techniques. First, arranging the memories in 4 banks, or collections of independently cycleable memories achieves the equivalent of a 4-ported memory (e.g. a write may be done to banks Ø and 1 in the same clock cycle as a read from banks 2 and 3). Second, the present invention utilizes SRAMs having access times at least twice as fast as the system clock speed, effectively allowing two accesses to each bank during a given system clock cycle. In the preferred embodiment, the system clock runs at a 120 nanoseconds (ns), requiring use of SRAMs with an access time of at most 60 ns. It is preferred to utilize SRAMs with an access time of 35 ns or faster. Therefore, each of the four banks may be accessed two times during any given clock cycle, yielding a logical 8-ported memory.

It will be obvious to one of ordinary skill that these methods may be utilized either individually or in combination to achieve the objective of multi-ported memory. It will be further obvious that a different number of banks, different size memories or different speed memories achieving more than 2 accesses per system clock may be utilized without departure from the spirit and scope of the present invention.

Among the advantages offered by the present inventions use of vector register file comprised of SRAMs is the relative-low cost of SRAMs, availability of SRAMs from multiple vendors and the relative speed available from current SRAM technology. These

advantages and other factors allow use of a vector register file of a greater size than utilized in known vector processing systems.

The present invention discloses several inventive techniques and features for management and use of the vector register file.

As one feature of the present invention, each of the plurality of banks may be divided into a plurality of vector registers. For example, referring to Figure 2, each of the banks 201, 202, 203 and 204 may be divided into a plurality of vector registers such as vector registers 210, 211, 212 and 213. In the particular example in Figure 2, each 2,048-element bank is divided into 64 32-element vector registers.

In fact, the banks, 301, 302, 303 and 304 may be divided into any number n, of vector registers, up to a maximum of p, where a p is the number of elements in a bank. Each vector register has p/n elements. The particular values for p and n may be chosen by convention depending on the advantage and disadvantage tradeoffs of having a large number of vector registers versus having vector registers with a large number elements (i.e. depth) in a particular case. The particular convention may be enforced by either hardware apparatus or software methods.

Utilizing the present invention, a vector register may begin at any element in a bank. The particular method of addressing elements will be described in more detail with reference to Figure 6.

3NSDOCID: <GB___2216307A__I_>

In the preferred embodiment, a particular vector register may be referred to utilizing a bank number and an offset. For example VR0(0) may refer to the first vector register in banks 0, VR0(32) may refer to the second vector register in bank 0 where the vector registers are 32-elements long, VR0(64) may refer to the third vector register in bank 0, etc.

As one alternative, vector registers may be referred to utilizing a bank number and a vector number. For example, VR0,0 may refer to a vector register in bank 0, number 0. The system may then be responsible for translating the vector number (i.e. 0) to an offset. VR0,1 may refer to the second vector register in bank 0. Again, the system may be responsible for translating the vector number (i.e. 1) to an offset. For example, if each vector register is to be 32-elements wide, the system would translate vector number 1 to offset 32.

As a third alternative, vector registers may be referred to with a vector register number only. The system may then be responsible for translating the vector number to a bank and offset. For example, VR0 may be translated to bank 0, offset 0, VR1 may be translated to bank 1, offset 0, VR2 may be translated to bank 2, offset 0, VR3 may be translated to bank 3, offset 0, VR4 may be translated to bank 0, offset 32, etc.

Each of these alternatives may offer tradeoffs in portability of program code versus speed of translated the vector numbers.

VECTOR REGISTER CONTEXT AREAS

In the present invention, as previously discussed, a comparatively large amount of vector register space is available. It is generally not necessary to have available to any single process the entire amount of vector register space. Therefore, as another inventive aspect of the present invention, vector register space may be divided into a plurality of process context areas. The system may then support a plurality of processes without the requirement of swapping a process' context data from the vector register file to another storage medium, such as a disk, when switching between processes.

Referring to Figure 3, in one embodiment, the 2,048-element banks 301, 302, 303 and 304 are divided into eight context areas 310-317 of length 256 elements in each bank.

Referring to Figure 4, when running a process, the system maintains a base register, SBASE 410. SBASE 410 contains the offset of the current context area from the beginning of the bank (i.e. from element 0). For example, if the current context is context area 312 of Figure 3, SBASE 410 contains the value 512. Utilizing SBASE 410, the system may switch between the current process and a process with context data in a different context area by changing the value of SBASE 410. When running a process, the value of SBASE 410 is added to the offset value of any reference to vector memory.

The SBASE register is further utilized to insure user processes do not reference offsets below the

NSDOCID: <GB___2216307A__I_>

assigned context area. Further, use of the SBASE register allows a process to be written independent of the particular context area it is assigned to. A process may be written to run as if it would always execute in the first context 310.

It will be obvious to one of ordinary skill that the vector register file may be divided into any number of context areas n between 1 and p, where p is the number of elements in a bank. The number of elements per context area is equal to p/n. The particular number n may be determined by a number of factors. It will be further obvious that it is possible to vary the size of the context areas, such that they are not of equal size, allowing use of a smaller context area for a relatively small process and a larger context area for a relatively large process.

The present invention further utilizes a second register, SLIMIT 411, to indicate the last referenceable element of the current context area. SLIMIT 411 is used to insure a process does not attempt to reference an element at an offset greater than an offset in its context area. Any reference to an element in the vector register file is compared against SLIMIT 411. If the reference is for an element with an offset greater than SLIMIT 411, it is inhibited and an error condition is indicated.

The present invention further utilizes a third register, UBASE 412, which allows a particular process to further divide its own context space. Typically, the operating system may be responsible for manipulating the

SBASE and SLIMIT registers. A user process may manipulate the UBASE register. For example, a process may have context data for individual subroutines located at fixed offsets from a given value for UBASE 412. The user sets UBASE 412 when executing the subroutine tc the given value for UBASE 412. The subroutine then may reference elements in the vector register file by supplying the offset value from UBASE 412. The system computes the actual offset within the vector register file by computing (SBASE 410) + (UBASE 412) + (the given offset).

Referring to Figure 5, a method for sharing context data and subroutines is disclosed. A process may locate its context data anywhere within the limits of SBASE 510 and SLIMIT 511. The process may initially set its UBASE register to point 520 to reference area 530 when running a first subroutine. The process may change the UBASE register to point 521 to reference area 531. Area 540 contains shared variables or other shared context information which may be referenced by either subroutine.

A second use of the UBASE register is to allow addressing of the full system base area (area bounded by SBASE and SLIMIT) using a limited length address field. For example, assume the system base area is 512 elements long and an eight bit address is to be used for addressing within this area. To address elements with relative offsets from SBASE of 0 to 255, UBASE is set to 0. To address elements with relative offsets from SBASE

3NSDOCID: <GB___2216307A__I_>

of 256 to 511, UBASE is set to 256 and elements are addressed as (0 to 255) + UBASE (i.e. 256 to 511).

Referring now to Figure 6, an operand descriptor 600 for instructions of the preferred embodiment is described. Instructions in the preferred embodiment may comprise three operands; vector register x (VR_X), 601(a) and 601(b), vector register y (VR_Y), 602(a) and 602(b), and vector register z (VR_Z), 603(a) and 603(b).

A vector register may be addressed by supplying a bank number in bits 9 and 8 of the vector register operand and an offset value in bits 7-0.

In the preferred embodiment, bits 9 through 5 of each vector register operand are placed in the high order half of the word 610. Utilizing bits 9 through 5 of the address, any of four banks of the preferred embodiment may be addressed with bits 9 and 8, and offsets within the banks may be addressed in multiples of 32. For example, a value of 00001_2 in bits 9 through 5 of a vector register operand would reference bank 0, offset 32. A value of 11010_2 would reference bank 3, offset 64. As discussed previously, the values of SBASE and UBASE are added to the offsets.

The splitting of the vector register address allows a 16-bit constant, instead of a 32-bit constant, to reference a vector register. It will be obvious to one of ordinary skill that the particular address bit organization may be modified without departure from the scope and spirit of the present invention.

In the preferred embodiment, an immediate constant of 16 bits or less may be stored as part of an

instruction. Therefore, by utilizing only the high-order half of a word to store operand information, the present invention is able to store the operand information in the instruction. This method saves processing time due to not having to retrieve the operand information separate from the instruction.

If it is desired to address a cell of a vector register directly or utilize a vector register for storing scalar variables, the lower half of the operand word 611 provides the remainder of the offset address information in bits 4 through 0 of the offset. It is also required to utilize bits 4 through 0 of the offset in order to allow for overlapping data for subroutines as described previously in connection with Figure 5.

Cell-level addressability allows for convolution and recurrence calculation to be performed. For example, utilizing the cell-level addressability feature, a convolution calculation such as:

```
DO  I = 1, 32
 Y(I) = W(1) * X[I+1]
      + W(2) * X[I+2]
      + W(3) * X[I+3]
END
```

may be calculated. Further, the preferred embodiment of the present invention executes all operations, including operations on individual elements of a vector, in an order consistent with sequential execution. This, in combination, with cell-level addressability allows

3NSDOCID <GB___2216307A__I_>

recurrences, such as a Fibonacci sequence, to be calculated directly.

Thus, a vector register file and method for managing access to the vector register file is described.

CLAIMS

1. A vector register file for a vector processing computer system, comprising;

a first bank of memory circuits;

a second bank of memory circuits, said second bank of memory circuits cycleable independent from said first bank.

2. A vector register file, as recited by Claim 1, wherein said first bank of memory circuits is comprised of static random access memories.

3. A vector register file, as recited by Claim 1, wherein said first bank of memory circuits is comprised of eight 2,048 by 8 static random access memories.

4. A vector register file, as recited by Claim 1, further comprising:

a third bank of memories, said third bank of memories cycleable independent from said first bank of memories and said second bank of memories;

a fourth bank of memory circuits, said fourth bank of memory circuits cycleable independent from said first bank of memory circuits, said second bank of memory circuits, and said third bank of memory circuits.

5. A vector register file, as recited by Claim 1, wherein each of said memory circuits is accessible at 2 times the system clock cycle time.

3NSDOCID <GB___2216307A__I_>

6. A vector register file, as recited by Claim 5, wherein said system clock cycle time is approximately 120 ns and said memory circuits may be accessed at most approximately 60 ns.

7. A vector register file, as recited by Claim 5, wherein said system clock cycle time is approximately 120 ns and said memory circuits may be accessed at least approximately 35 ns.

8. A vector processing computer system having a vector register file, comprising:

a system clock having a cycle time of n;

a plurality of memory circuits, said memory circuits having an access time of at least approximately n/p, where p is at least 2.

9. The vector processing computer system, as recited by Claim 8, wherein n is approximately 120 ns and p is at most approximately 60 ns.

10. A vector processing computer system, comprising:

a vector register file;

a system base register for storing a system base value, said system base value being added to an offset value for addressing said vector register file.

11. A vector processing computer system, as recited by Claim 10, further comprising a limit register for storing a limit value, said limit value being compared with the sum of said system base value and said offset, if said sum is greater than said limit, an error condition being indicated.

12. A vector processing computer system, as recited by Claim 10, further comprising a user base register for storing a user base value, said user base value being added to said offset value and said system base value when addressing said vector register file.

13. A vector processing computer system, as recited by Claim 10, wherein said system base value is changed when said vector processing computer system changes active processes.

14. In a vector processing computer system having a vector register file and a system base register, said system base register for storing a system base value, a method for addressing vector register file comprising the steps of:

(a) adding an offset value to said system base value giving a sum;

(b) addressing said vector register file at said sum; and

(c) said vector processing computer system changing its active process;

BNSDOCID <GB__2216307A__I_>

(d) changing the value of said system base register in response to said changing of processes.

15. The vector processing computer system, as recited by Claim 14, further comprising a user base register for storing a user base value said method further comprising the step of adding said user base value to said sum prior to addressing said vector register file.

16. The vector processing computer system, as recited by Claim 14, further comprising a limit register for storing a limit value, said method further comprising the step of comparing said sum with said limit value and generating an error condition if said sum is greater than said limit value prior to addressing said vector register file.

17. The vector processing computer system, as recited by Claim 16, further comprising the step of changing said limit value in response to said changing of processes.

18. In a vector processing computer system having a vector register file, said vector register file comprising addressable elements, a method of dividing said vector register file into a plurality of vector registers comprising the steps of:

(a) determining a value n, said value n indicating a desired number of vector registers;

(b) logically dividing said vector register file into said plurality of vector registers, each of said vector registers having a depth of p/n elements.

19. The vector processing computer system of Claim 18, wherein p is 2048.

20. In a vector processing computer system having a vector register file, said vector processing computer system having instructions requiring a plurality of operands, said operands comprising addresses within said vector register file, a method of storing said operands, comprising:

storing a first plurality of bits of said address for each of said operands in a first half of a word;

storing a second plurality of bits of said address of each of said operands in a second half of said word.

21. The vector processing computer system, as recited by Claim 20, wherein said first plurality of bits comprises bits 9 through 5 of said address of each of said operands.

22. The vector processing computer system, as recited by Claim 21, wherein said second plurality of bits comprises bits 4 through Ø of said address of each of said operands.

23. A vector register file for a vector processing computer system substantially as hereinbefore described with reference to the accompanying drawings.

BNSDOCID: <GB___2216307A__I_>

Amendments to the claims
have been filed as follows

1. A vector register file for a vector processing computer system, comprising:

a first bank of memory circuits for storing a first plurality of vectors;

a second bank of memory circuits for storing a second plurality of vectors, said second bank of memory circuits cycleable independent from said first bank.

2. A vector register file, as recited by Claim 1, wherein said first bank of memory circuits is comprised of static random access memories.

3. A vector register file, as recited by Claim 1, wherein said first bank of memory circuits is comprised of eight 2,048 by 8 static random access memories.

4. A vector register file, as recited by Claim 1, further comprising:

a third bank of memories, said third bank of memories cycleable independent from said first bank of memories and said second bank of memories;

a fourth bank of memory circuits, said fourth bank of memory circuits cycleable independent from said first bank of memory circuits, said second bank of memory circuits, and said third bank of memory circuits.

5. A vector register file, as recited by Claim 1, wherein each of said memory circuits is accessible at 2 times the system clock cycle time.

6. A vector register file, as recited by Claim 5, wherein said system clock cycle time is approximately 120 ns and said memory circuits may be accessed at most approximately 60 ns.

7. A vector register file, as recited by Claim 5, wherein said system clock cycle time is approximately 120 ns and said memory circuits may be accessed at least approximately 35 ns.

8. A vector processing computer system having a vector register file, comprising:

a system clock having a cycle time of n;

a plurality of memory circuits, said memory circuits having an access time of at least approximately n/p, where p is at least 2.

9. The vector processing computer system, as recited by Claim 8, wherein n is approximately 120 ns and p is at most approximately 60 ns.

10. A vector processing computer system, comprising:

a vector register file;

a system base register for storing a system base value, said system base value being added to an offset value for addressing said vector register file.

BNSDOCID: <GB___2216307A_I_>

-47-

11. A vector processing computer system, as recited by Claim 10, further comprising a limit register for storing a limit value, said limit value being compared with the sum of said system base value and said offset, if said sum is greater than said limit, an error condition being indicated.

12. A vector processing computer system, as recited by Claim 10, further comprising a user base register for storing a user base value, said user base value being added to said offset value and said system base value when addressing said vector register file.

13. A vector processing computer system, as recited by Claim 10, wherein said system base value is changed when said vector processing computer system changes active processes.

14. In a vector processing computer system having a vector register file and a system base register, said system base register for storing a system base value, a method for addressing vector register file comprising the steps of:

(a) adding an offset value to said system base value giving a sum;

(b) addressing said vector register file at said sum; and

(c) said vector processing computer system changing its active process;

(d) changing the value of said system base register in response to said changing of processes.

15. The vector processing computer system, as recited by Claim 14, further comprising a user base register for storing a user base value said method further comprising the step of adding said user base value to said sum prior to addressing said vector register file.

16. The vector processing computer system, as recited by Claim 14, further comprising a limit register for storing a limit value, said method further comprising the step of comparing said sum with said limit value and generating an error condition if said sum is greater than said limit value prior to addressing said vector register file.

17. The vector processing computer system, as recited by Claim 16, further comprising the step of changing said limit value in response to said changing of processes.

18. In a vector processing computer system having a vector register file, said vector register file comprising addressable elements, a method of dividing said vector register file into a plurality of vector registers comprising the steps of:

(a) determining a value n, said value n indicating a desired number of vector registers;

NSDOCID <GB___2216307A__I_>

(b) logically dividing said vector register file into said plurality of vector registers, each of said vector registers having a depth of p/n elements.

19. The vector processing computer system of Claim 18, wherein p is 2048.

20. In a vector processing computer system having a vector register file, said vector processing computer system having instructions requiring a plurality of operands, said operands comprising addresses within said vector register file, a method of storing said operands, comprising:

storing a first plurality of bits of said address for each of said operands in a first half of a word;

storing a second plurality of bits of said address of each of said operands in a second half of said word.

21. The vector processing computer system, as recited by Claim 20, wherein said first plurality of bits comprises bits 9 through 5 of said address of each of said operands.

22. The vector processing computer system, as recited by Claim 21, wherein said second plurality of bits comprises bits 4 through 0 of said address of each of said operands.

23. A vector register file for a vector processing computer system substantially as hereinbefore described with reference to the accompanying drawings.

THIS PAGE BLANK (USPTO)

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

This Page Blank (usptc)